



JOHN DEERE



Fraunhofer
IESE

Agile Doesn't Scale ... Without Architecture

Brian Cronauer, John Deere
Andreas Huber, John Deere
Michael Höh, John Deere
Thorsten Keuler, Fraunhofer IESE

SATURN 2012
St. Petersburg, FL
May 10, 2012



architecture.iese.fraunhofer.de

Starting Point

Transition towards Agile

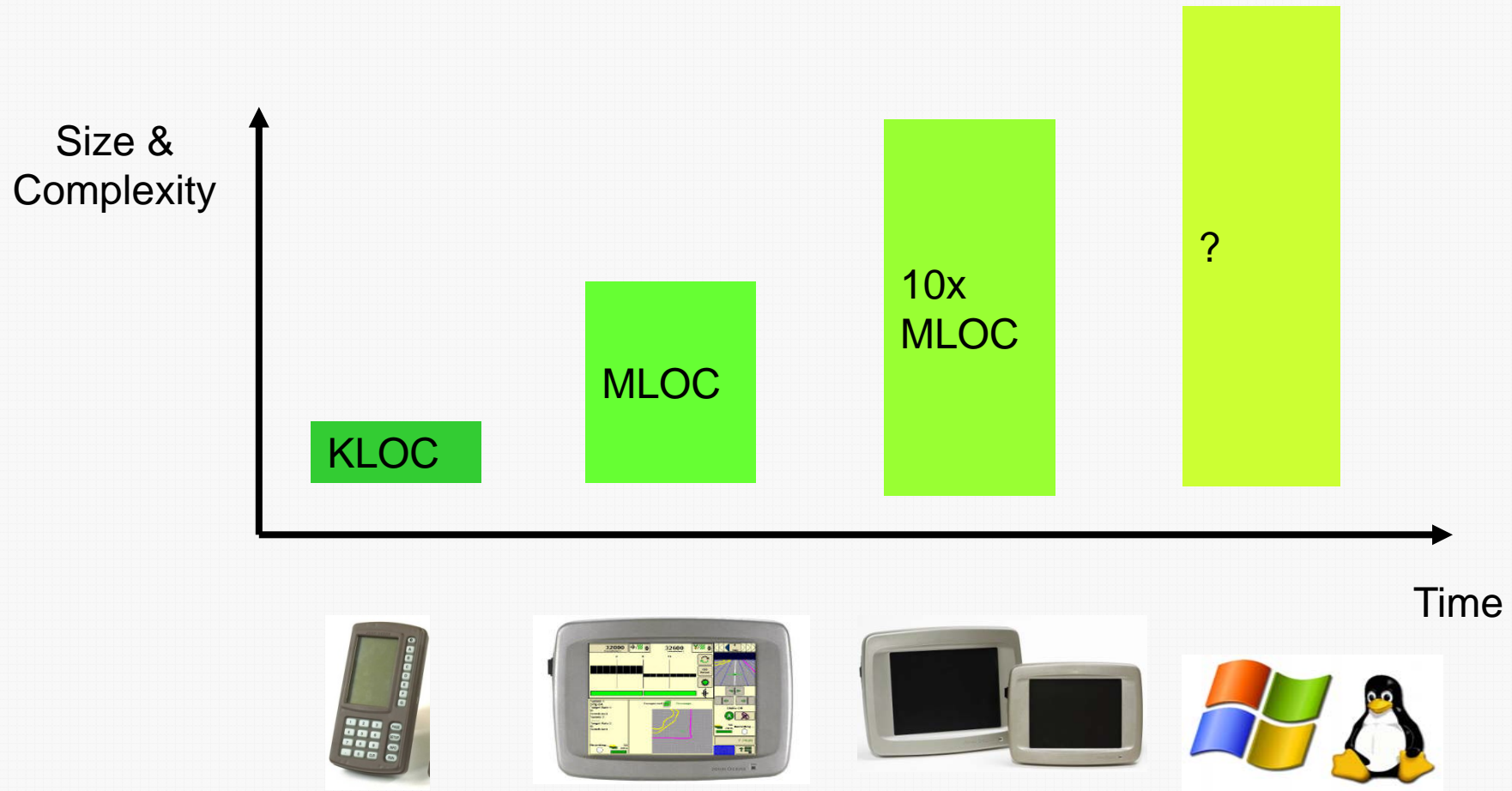
Scalability of SW Development

Experiences

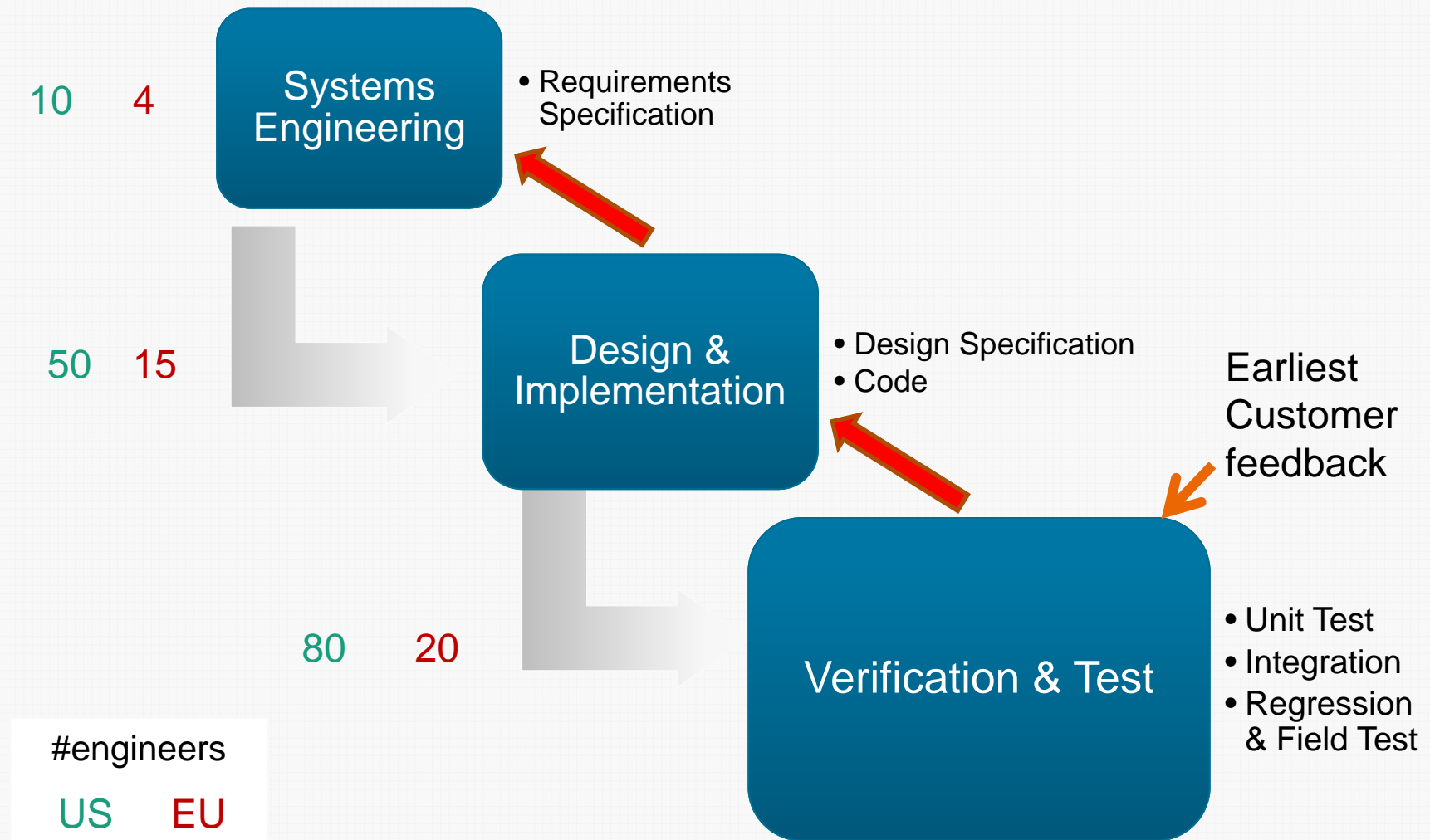
What have we learned?



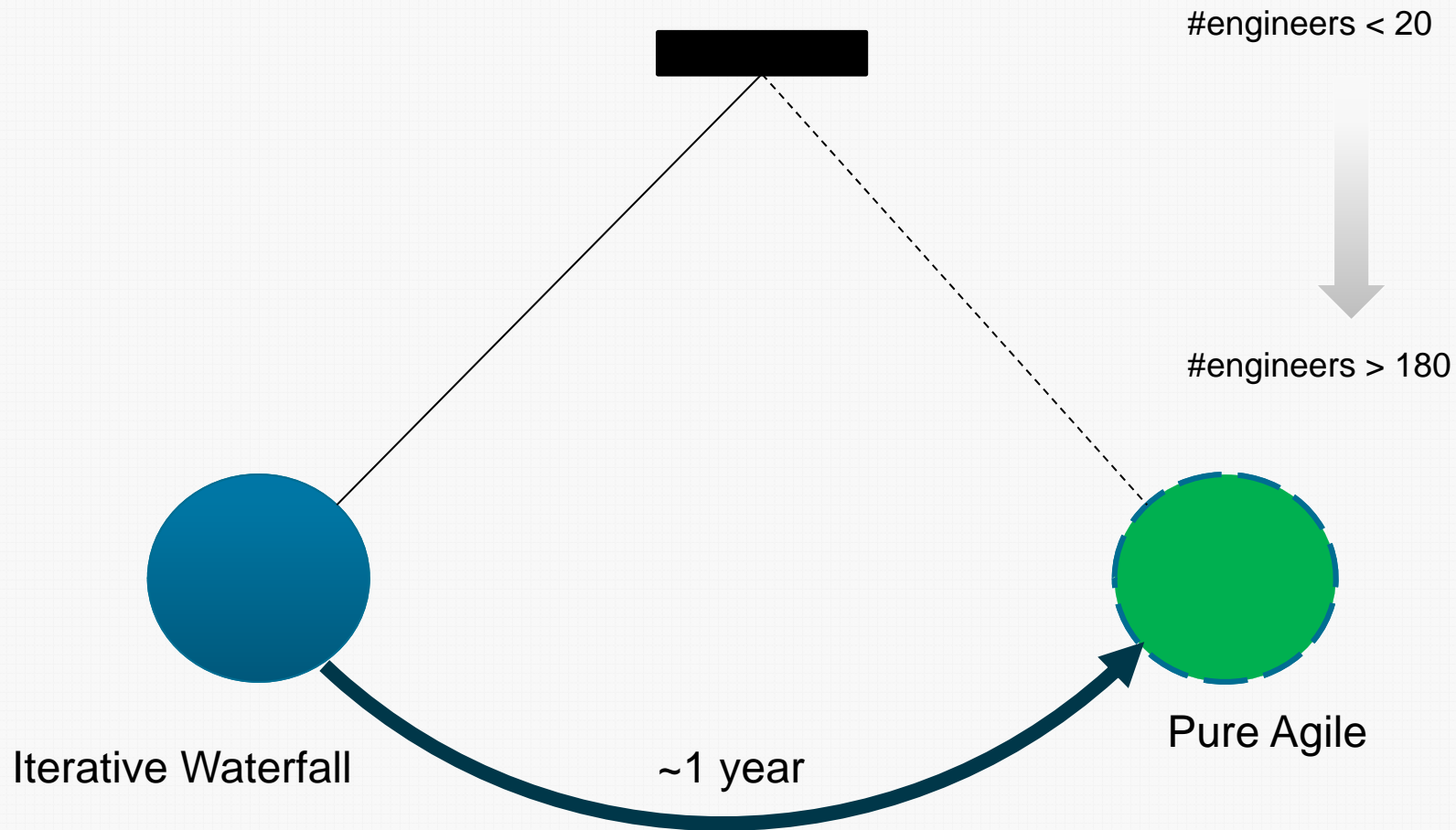
John Deere – Display Development



Starting Point



Large-Scale Transition Towards Agile



Starting Point

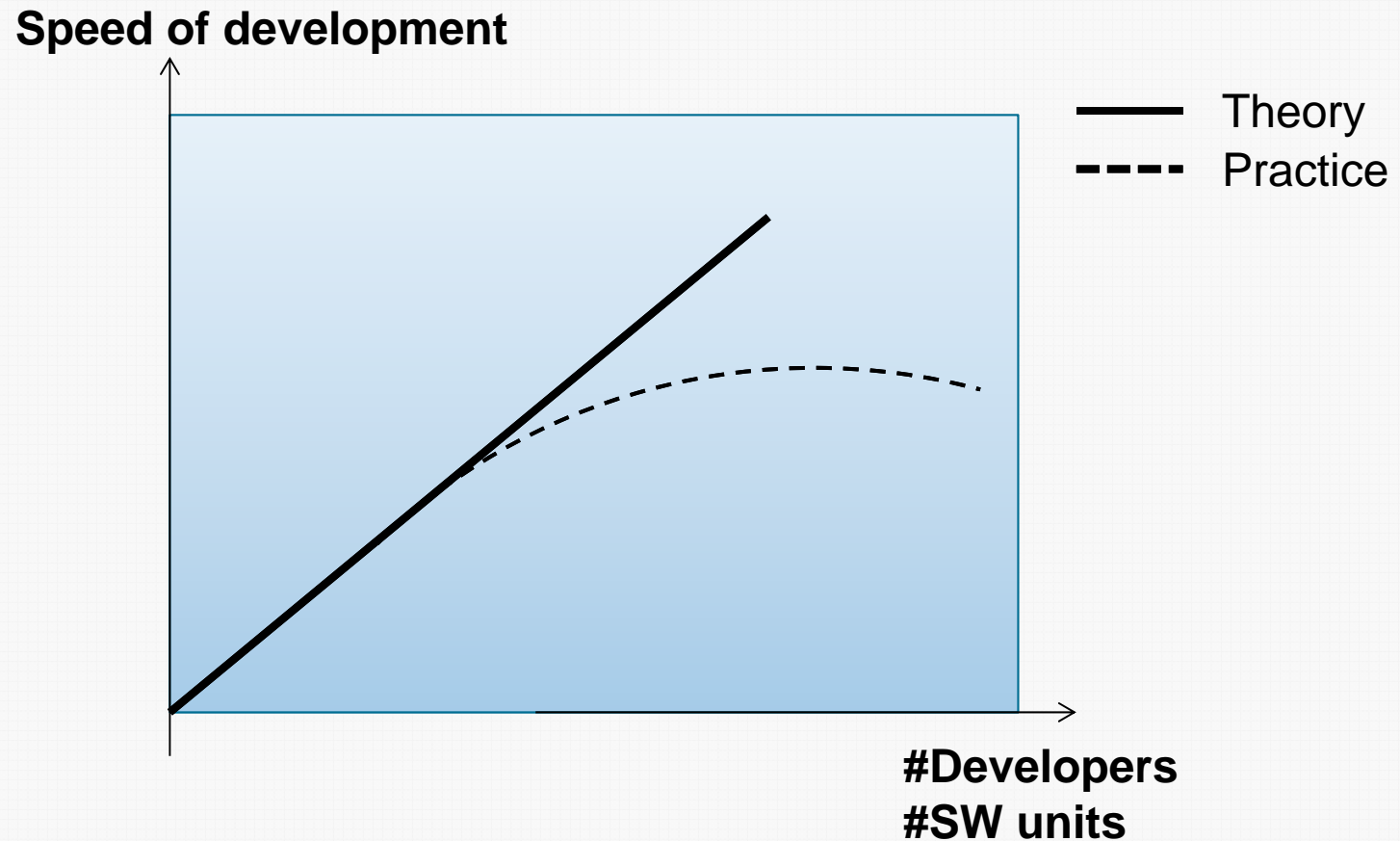
Transition towards Agile

Scalability of SW Development

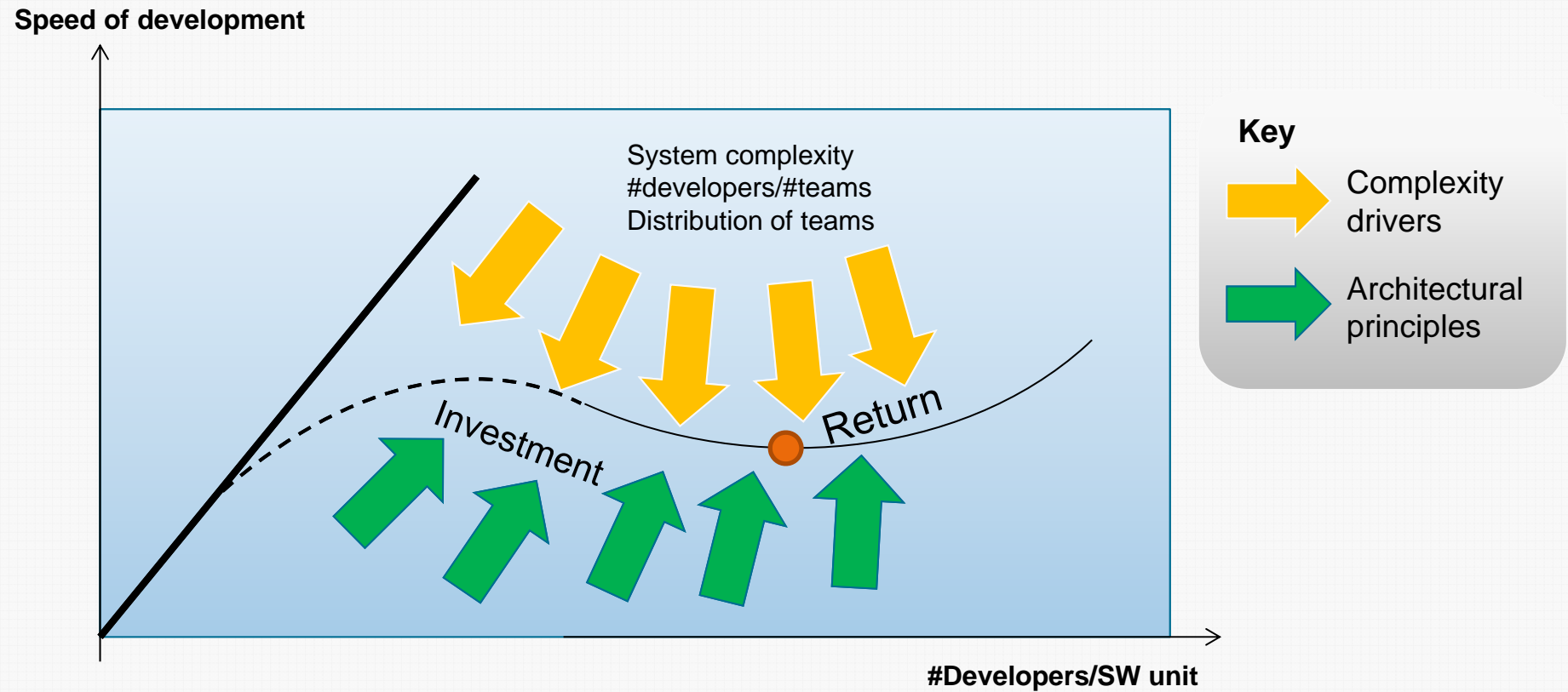
Experiences

What have we learned?

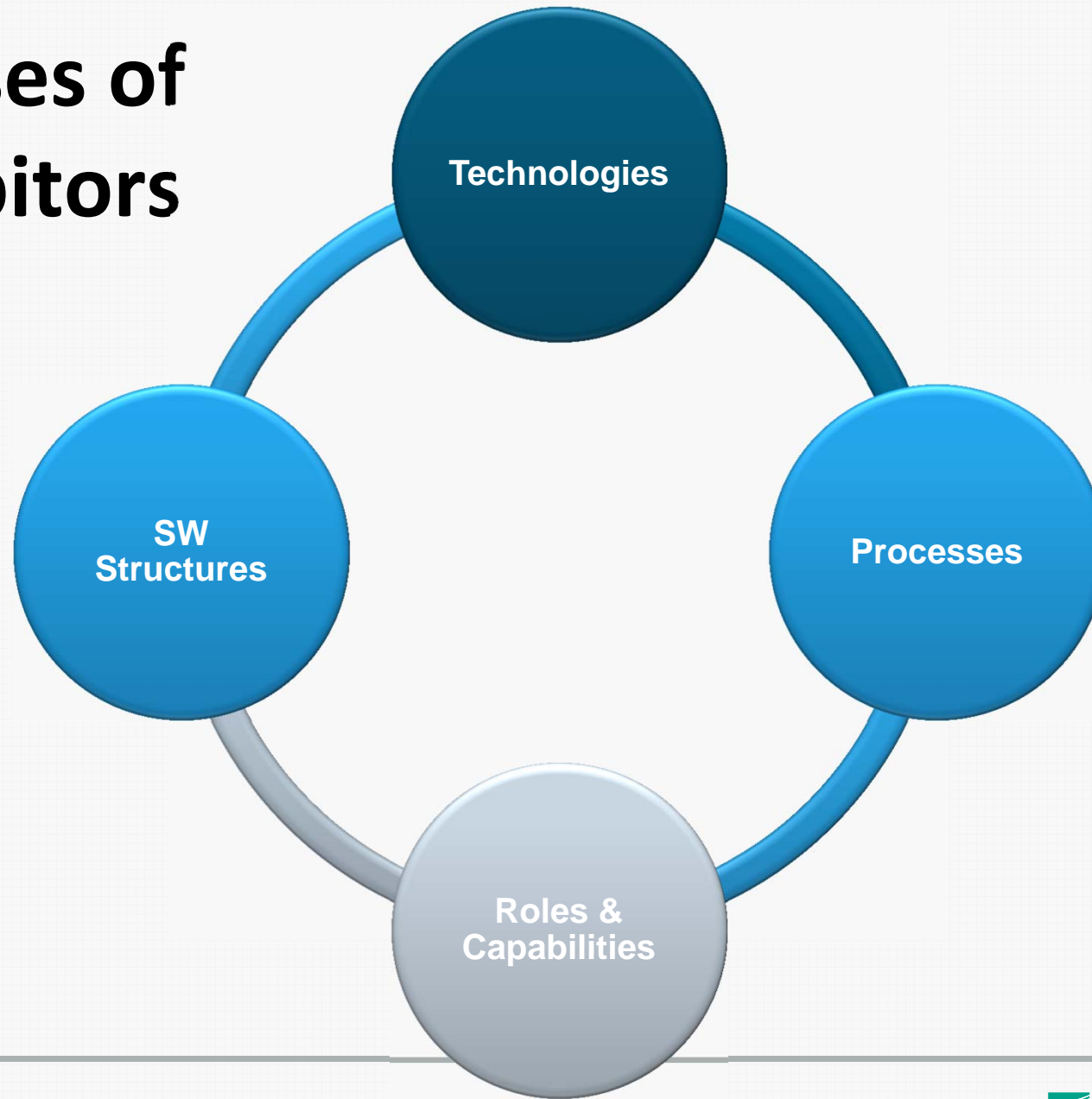
What is the Scale in „Large-Scale“?



Inhibitors to Scalability



Classes of Inhibitors



Starting Point

Transition towards Agile

Scalability of SW Development

Experiences

What have we learned?

Transition to Agile

Effects

Prototyping with one team

worked well with one team

faster delivery

Reasons

Small scale

Co-located developers

Small system scope

“Power of the Team”

Effects

communication overhead

Coding guidelines not followed

Java vs C++

Deletion of work

Reasons

Clear phases are gone

Specs -> user stories

Rules & standards?

„Do what you want“

**“With great Power comes
great Responsibility”**

THE

A - TEAM

Communication vs Documentation

Effects

... we don't document

... we don't do architecture

... we don't think about next PSI

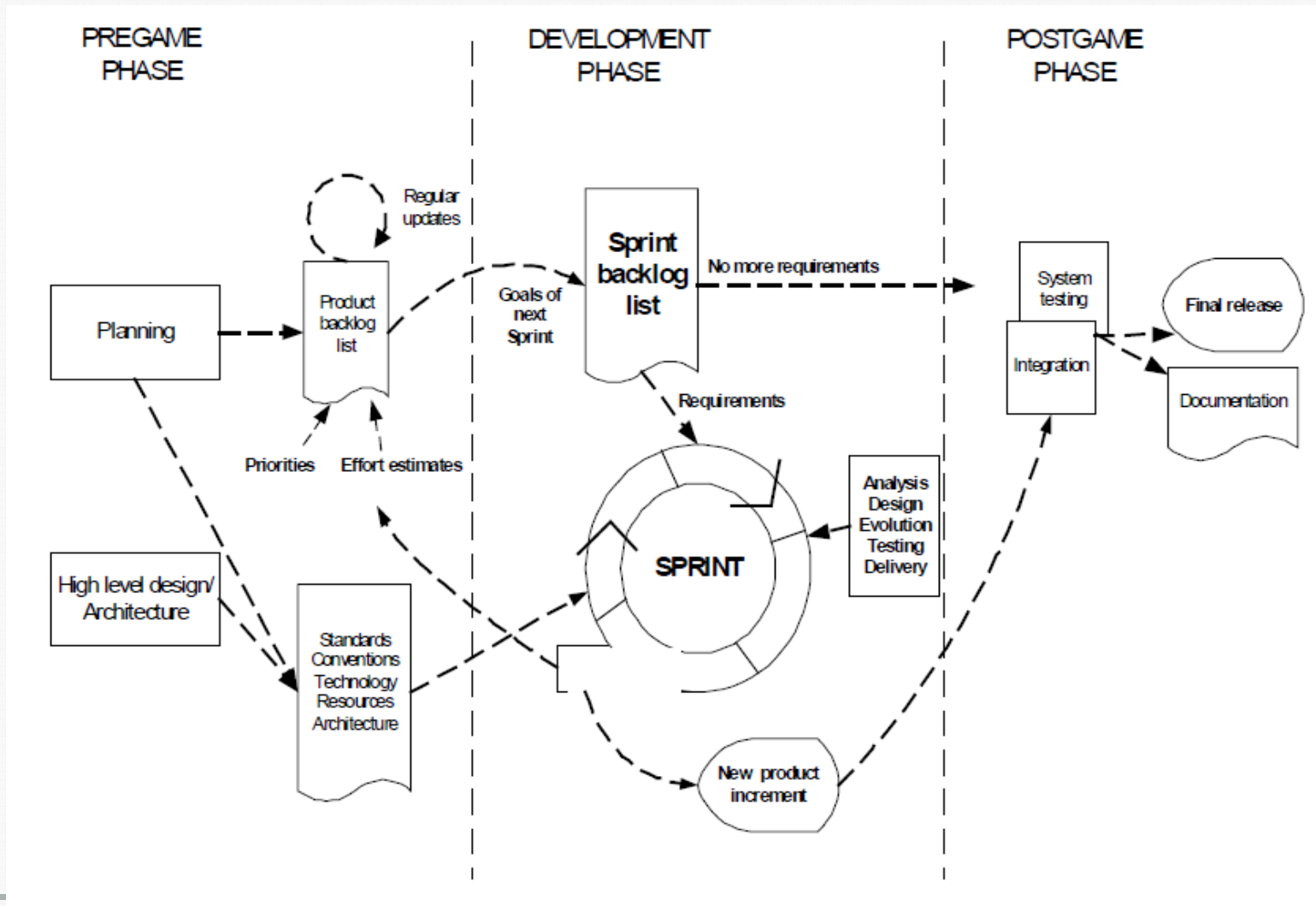
... YAGNI

Reasons

Agile ∩ Agile Techniques

Philosophy Clashes

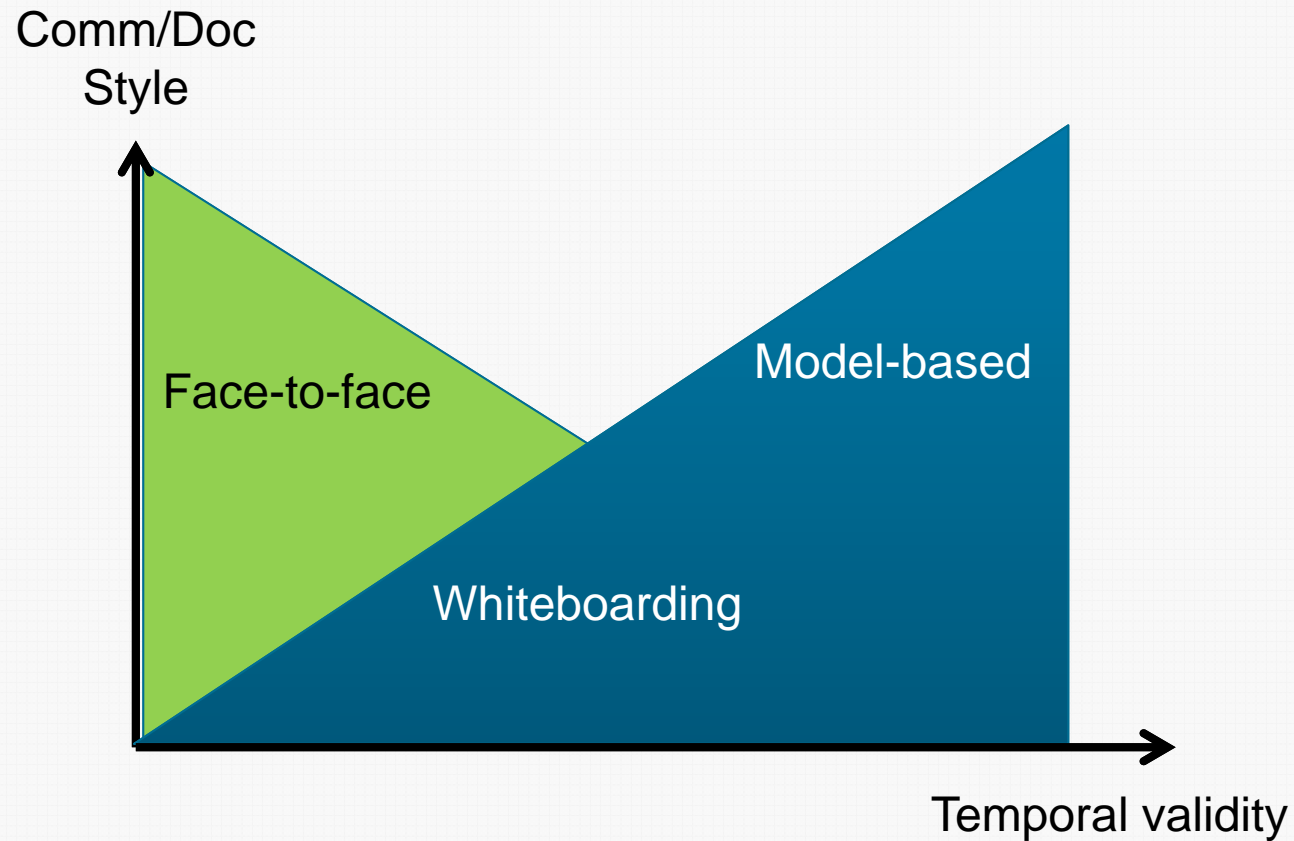
Philosophies – Revisited



„Disciplined agile teams ...“

- ... work closely with their **stakeholders**, including both operations and support staff
- ... invest time at the beginning of the project to identify the high-level **scope** in a light-weight, collaborative manner
- ... will also identify a viable **architectural strategy** which reflects the **requirements** of their stakeholders and your organization's overall architectural strategy

Comm/Doc – Rule of Thumb



Requirements

Effects

Late Detection of Quality issues

Incompatible solutions for
crosscutting concerns

Reasons

NF requirements implicit

Big Picture?

Feature-based development

“Tooling War”

Effects

Replaceability of teams

Exchangeability of artifacts

Reasons

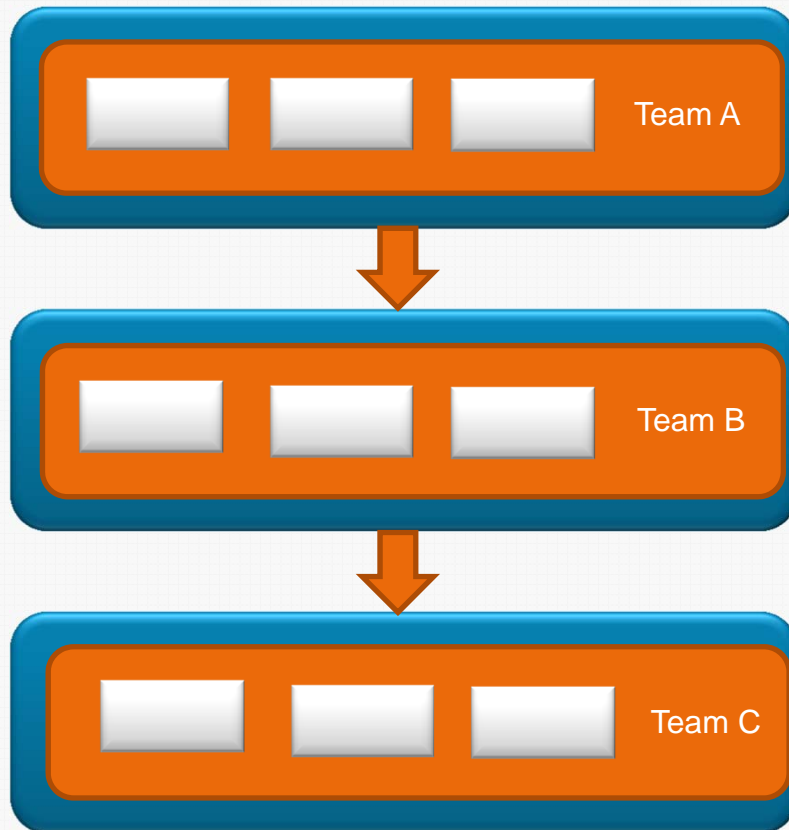
No rules what tools to use

No decision makers

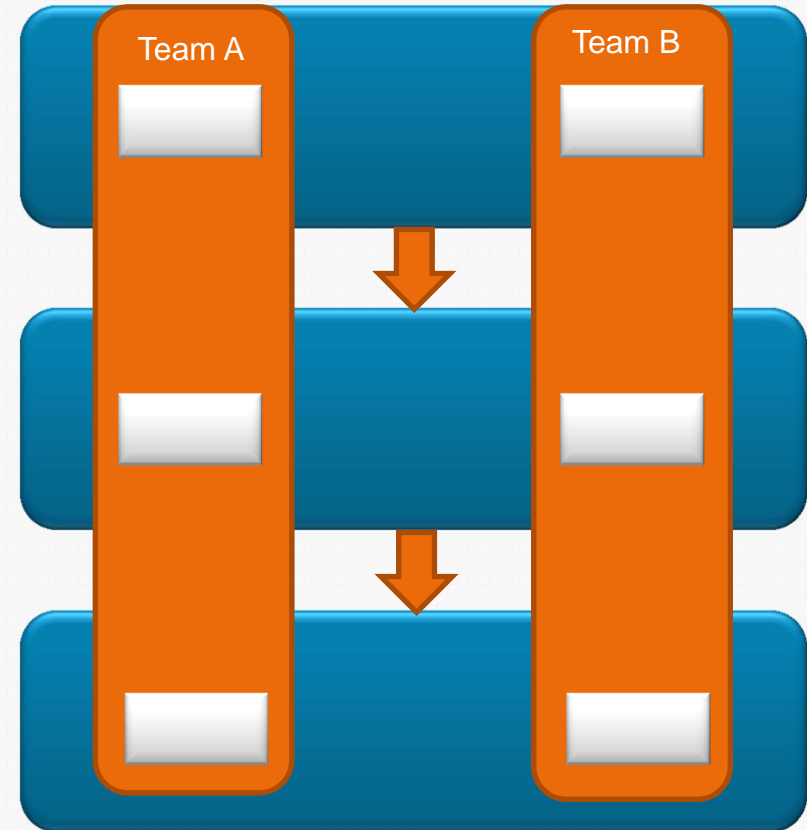
No training

Vertical Slices Or Features vs Components

Component-based vs Feature-based



Component-based



Feature-based

Effects

Changing components concurrently

Unintended dependencies

Redundant implementations

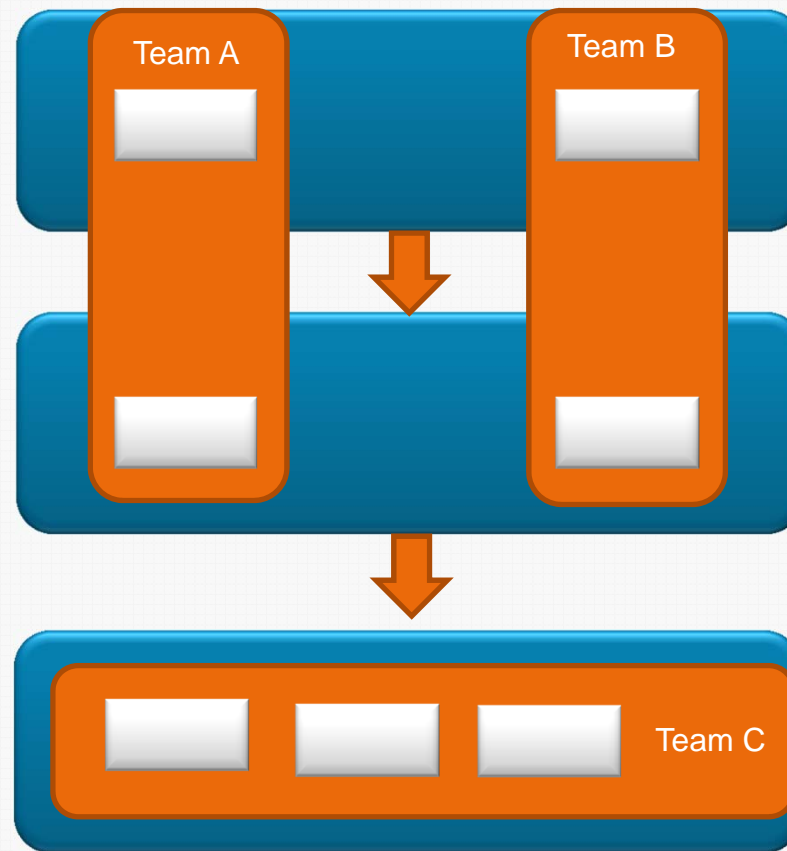
Reasons

Little domain expertise

Unclear/No Responsibilities?

No Alignment/Synchronization

Feature-based & Component-based



**“Decoupled code for
decoupled teams”**

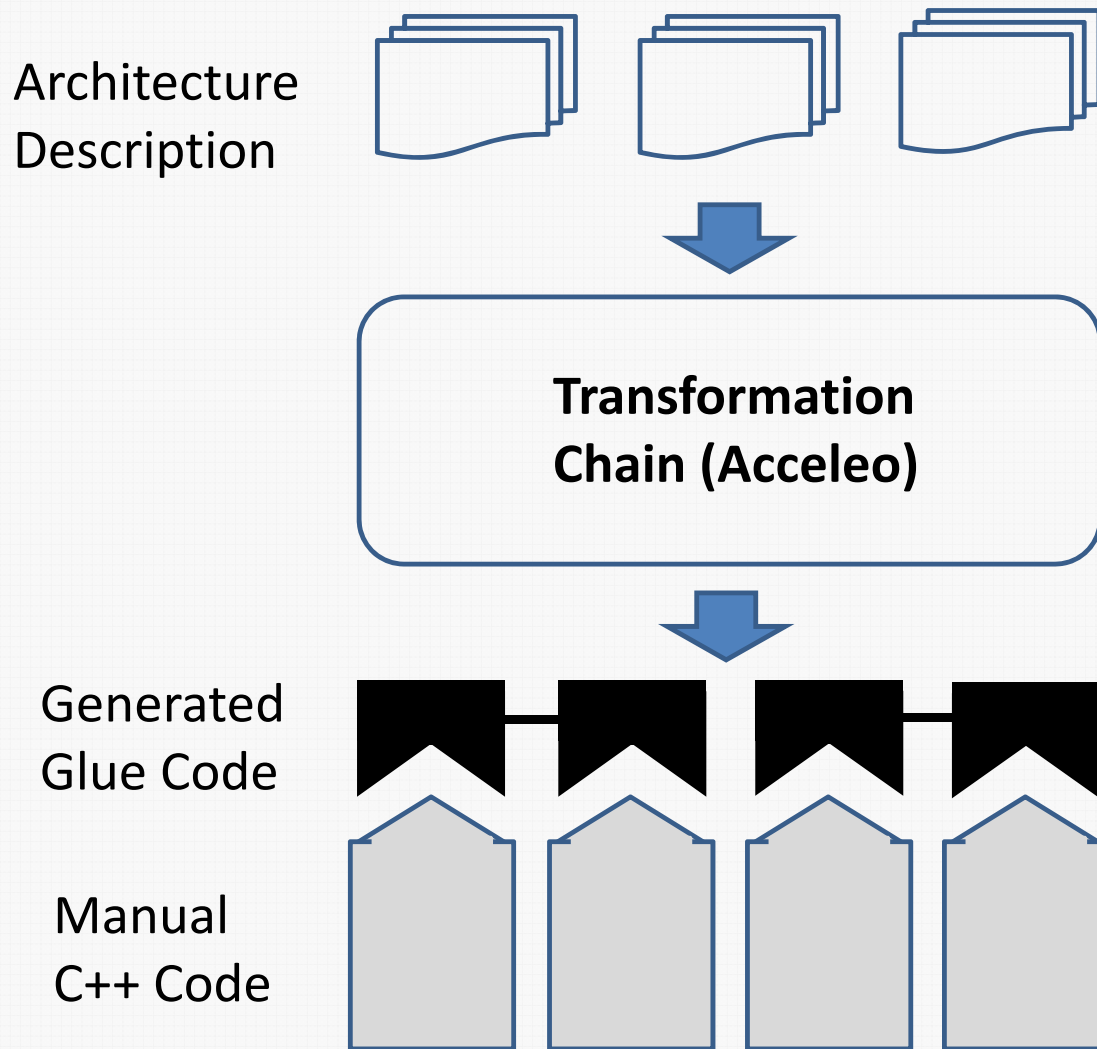
Dependency Control

Glue code generation

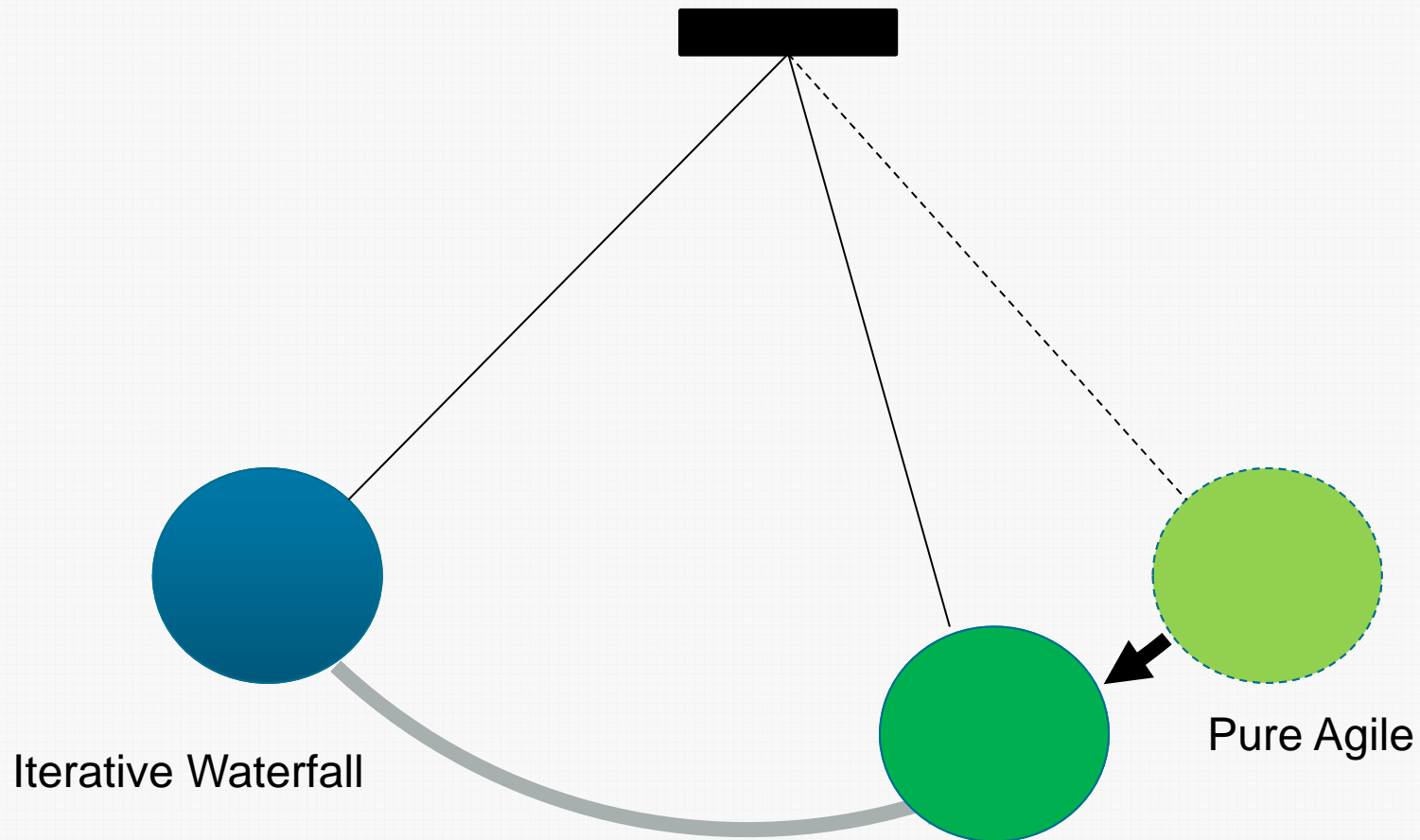
Unified implementations

Architecture enforcement

Common Technical Framework



Transition: Trend@Deere



Starting Point

Transition towards Agile

Scalability of SW Development

Experiences

What have we learned?

What have we learned?

■ Understand Change

- Classify change -> Is it architecturally-relevant?
 - Costly to change?
 - Impact on multiple teams?
 - Impact on many software parts?

■ Act on Change

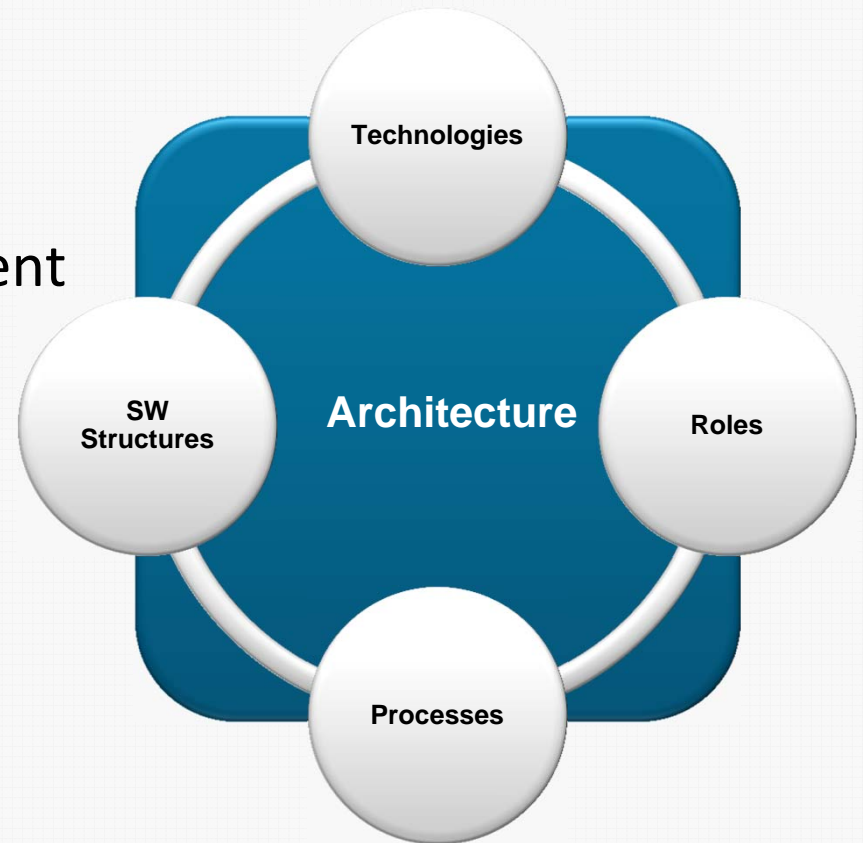
- Management commitment -> Who pays for “infrastructure” change?

■ Architecture and Agile are **not** Adversaries

- Combine the best of both worlds!

What have we learned?

- Unification
 - Solid basis for scaling development
 - Agreement on technologies
 - True commitment to architectural rules
- **Reasonable** design “look ahead”
 - Check scalability across inhibitors



**Thank you for your
attention!**

... Questions?



<http://architecture.iese.fraunhofer.de>

Contact

Dr. Thorsten Keuler

Phone: +49 631 6800 2162

thorsten.keuler@iese.fraunhofer.de

<http://architecture.iese.fraunhofer.de>